

# Package ‘MEDME’

July 13, 2025

**Type** Package

**Title** Modelling Experimental Data from MeDIP Enrichment

**Version** 1.69.0

**Date** 2009-3-26

**Author** Mattia Pelizzola and Annette Molinaro

**Description** MEDME allows the prediction of absolute and relative methylation levels based on measures obtained by MeDIP-microarray experiments

**Maintainer** Mattia Pelizzola <mattia.pelizzola@gmail.com>

**Depends** R (>= 2.15), grDevices, graphics, methods, stats, utils

**Imports** Biostrings, MASS, drc

**Suggests** BSgenome.Hsapiens.UCSC.hg18, BSgenome.Mmusculus.UCSC.mm9

**License** GPL (>= 2)

**LazyLoad** yes

**biocViews** Microarray, CpGIsland, DNAMethylation

**git\_url** <https://git.bioconductor.org/packages/MEDME>

**git\_branch** devel

**git\_last\_commit** 8234ba6

**git\_last\_commit\_date** 2025-04-15

**Repository** Bioconductor 3.22

**Date/Publication** 2025-07-13

## Contents

CGcount . . . . .	2
MEDME . . . . .	3
MEDME.predict . . . . .	4
MEDME.readFiles . . . . .	5
MEDME.writeFiles . . . . .	5
MEDMEset-class . . . . .	6
smooth . . . . .	7
testMEDMEset . . . . .	8
<b>Index</b>	<b>9</b>

---

CGcount	<i>Determining the count of CpG dinucleotides for a set of genomic locations</i>
---------	--

---

## Description

The count of CpGs is determined in each window of size `wsiz`, with or without weighting, for each probe according to its position, chromosome and genome release

## Usage

```
CGcount(data, wsiz = 1000, wFunction = "linear")
```

## Arguments

<code>data</code>	An object of class <code>MEDMEset</code>
<code>wsiz</code>	number; the size of the smoothing window, in bp
<code>wFunction</code>	string; the type of weighting function, to choose among linear, exp, log or none

## Details

Only human and mouse are currently supported. The respective genomic sequence metadata library needs to be downloaded from the Bioconductor website, installed and loaded (around 800Mb). Please note that only the last genome release should be used. LiftOver UCSC tool could be used for batch conversion of old genomic position to the last genome release.

## Value

An object of class `MEDMEset` is returned where the count of CpGs for each probe has been saved on the `CGcount` slot.

## See Also

[smooth](#)

## Examples

```
data(testMEDMEset)
## just an example with the first 1000 probes
testMEDMEset = smooth(data = testMEDMEset[1:1000,])
library(BSgenome.Hsapiens.UCSC.hg18)
testMEDMEset = CGcount(data = testMEDMEset)
```

---

MEDME	<i>Determining the logistic model of MeDIP enrichment in respect to the expected DNA methylation level</i>
-------	--

---

## Description

Probe-level MeDIP weighted enrichment is compared to the expected DNA methylation level. The former is determined applying MeDIP protocol to a fully methylated DNA. The latter is determined as the count of CpGs for each probe. This is assumed to be the methylation level of each probe in a fully methylated sample.

## Usage

```
MEDME(data, sample, CGcountThr = 1, figName = NULL)
```

## Arguments

data	An object of class MEDMEset
sample	Integer; the number of the sample to be used to fit the model, based on the order of samples in the smoothed slot
CGcountThr	number; the threshold to avoid modelling probes with really low methylation level, i.e. CpG count
figName	string; the name of the file reporting the model fitting

## Details

The model should be applied on calibration data containing MeDIP enrichment of fully methylated DNA, most likely artificially generated (see references). Nevertheless, in case chromosome or genome-wide human tiling arrays are used a regular sample could be used too. In fact, human genomic DNA is known to be hyper-methylated but in the promoter regions. Of course the performance of the method is expected to be somehow affected by this approximation.

## Value

The logistic model as returned from the `multdrc` function from the `drc` R library

## References

<http://genome.cshlp.org/cgi/content/abstract/gr.080721.108v1>

## See Also

[smooth](#), [CGcount](#)

## Examples

```
data(testMEDMEset)
## just an example with the first 1000 probes
testMEDMEset = smooth(data = testMEDMEset[1:1000, ])
library(BSgenome.Hsapiens.UCSC.hg18)
testMEDMEset = CGcount(data = testMEDMEset)
MEDMEmodel = MEDME(data = testMEDMEset, sample = 1, CGcountThr = 1, figName = NULL)
```

MEDME.predict

*Applying the logistic model on MeDIP enrichment data***Description**

This allows the probe-level determination of MeDIP smoothed data, as well as absolute and relative methylation levels (AMS and RMS respectively)

**Usage**

```
MEDME.predict(data, MEDMEfit, MEDMEextremes = c(1,32), wsize = 1000, wFunction='linear')
```

**Arguments**

data	An object of class MEDMEset
MEDMEfit	the model obtained from the MEDME.model function
MEDMEextremes	vector; the background and saturation values as determined by the fitting of the model on the calibration data
wsize	number; the size of the smoothing window, in bp
wFunction	string; the type of weighting function, to choose among linear, exp, log or none

**Value**

An object of class MEDMEset. The resulting smoothed data, the absolute and relative methylation score (AMS and RMS) are saved in the smoothed, AMS and RMS slots, respectively.

**References**

<http://genome.cshlp.org/cgi/content/abstract/gr.080721.108v1>

**See Also**

[smooth](#), [CGcount](#), [MEDME](#)

**Examples**

```
data(testMEDMEset)
## just an example with the first 1000 probes
testMEDMEset = smooth(data = testMEDMEset[1:1000, ])
library(BSgenome.Hsapiens.UCSC.hg18)
testMEDMEset = CGcount(data = testMEDMEset)
MEDMEmodel = MEDME(data = testMEDMEset, sample = 1, CGcountThr = 1, figName = NULL)
testMEDMEset = MEDME.predict(data = testMEDMEset, MEDMEfit = MEDMEmodel, MEDMEextremes = c(1,32), wsize = 1000,
```

---

MEDME.readFiles	<i>reading sgr or gff files for MEDME</i>
-----------------	---

---

### Description

allows to read sgr or gff files before submitting the data to MEDME analysis

### Usage

```
MEDME.readFiles(path = getwd(), files = NULL, format, organism)
```

### Arguments

path	string; the path where the files are stored; the current working directory is the default
files	vector; optional vector of file names
format	string; either sgr or gff to indicate the respective file formats
organism	string; either hsa or mmu for homo sapiens and mus musculus respectively

### Details

In case of GFF files (recommened), tab-delimited files with header are expected with following fields: chromosome, probe ids, start and stop chromosomal positions, and score are expected in columns 1, 3, 4, 5 and 6 respectively. Multiple files are also expected to be in the same order of rows.

In case of sgr files (GFF is the preferred format), tab-delimited files with no header and chr, chr positions and score are expected in columns 1, 2 and 3 respectively. Multiple files are also expected to be in the same order of rows.

### Value

An object of class MEDMEset. The column headers in the logR slot are determined from the file names. in case of SGR files the are not probe names and progressive numbers are used in place of them. In case of GFF files the probe names are determined from the 3rd column.

---

MEDME.writeFiles	<i>writeFiles sgr or gff files from MEDME output</i>
------------------	--

---

### Description

allows to write sgr or gff files after MEDME analysis

### Usage

```
MEDME.writeFiles(data, output, path = getwd(), format, featureLength = NULL)
```

**Arguments**

data	An object of class MEDMEset
output	string; the name of the data slot to be written on the disk, either logR, smoothed, AMS or RMS
path	string; the path where the files are stored; the current working directory is the default
format	string; either sgr or gff to indicate the respective file formats
featureLength	integer; in case of GFF file format the length of the features has to be provided to determine start and end positions

**Details**

One GFF or SGR file is provided for each sample of the data MEDMEset object.

In case of GFF files, tab-delimited files with header are provided with following fields for each probe: chromosome, empty field, probe ids, start and stop chromosomal positions, and score and empty fields.

In case of sgr files, tab-delimited files with no header and chr, chr positions and score are provided.

---

MEDMEset-class	<i>Class "MEDMEset"</i>
----------------	-------------------------

---

**Description**

This class is used in MEDME library to store MeDIP derived DNA-methylation estimates and to save further elaboration of these, in association with chromosomal and positional probe information

**Objects from the Class**

Objects can be created by calls of the form `new("MEDMEset", ...)`. This object could initially host the MeDIP normalized logRatio data, as returned by the `MEDME.readFiles` function. Afterwards, the same object is returned by most of the MEDME library function. Each time, a new slot is filled with additional data, as smoothed logR or Absolute/Relative Methylation Scores (AMS and RMS respectively). At the end of the analysis, usually after a call to the `MEDME.predict` function, the `MEDME.writeFiles` function can be used to generate SGR or GFF files from this object.

**Slots**

chr: Object of class "character" : the probe-level chromosome assignments  
pos: Object of class "numeric" : the probe-level genomic position  
logR: Object of class "matrix" : the probe-level un-transformed normalized MeDIP logRatios for each sample  
smoothed: Object of class "matrix" : the probe-level smoothed MeDIP logRatios for each sample  
AMS: Object of class "matrix" : the probe-level Absolute Methylation Score for each sample  
RMS: Object of class "matrix" : the probe-level Relative Methylation Score for each sample  
CGcounts: Object of class "numeric" : the probe-level count of CpGs  
organism: Object of class "character" : the organism that the probe genomic positions are referring to, either hsa or mmu for homo sapiens or mus musculus respectively

## Methods

[ signature(x = "MEDMEset"): subsets the object based on its probes and/or samples  
**AMS** signature(object = "MEDMEset"): extracts the Absolute Methylation Score from the AMS slot  
**CG** signature(object = "MEDMEset"): extracts the probe CpG count from the CGcounts slot  
**chr** signature(object = "MEDMEset"): extracts the probe chromosomal assignment  
**org** signature(object = "MEDMEset"): extracts the organism  
**initialize** signature(.Object = "MEDMEset"): automatically generates smoothed, AMS and RMS matrix when only the logR slot is filled  
**logR** signature(object = "MEDMEset"): extracts the matrix of MeDIP un-transformed logRatios  
**pos** signature(object = "MEDMEset"): extracts the probe genomic position  
**RMS** signature(object = "MEDMEset"): extracts the Relative Methylation Score from the RMS slot  
**show** signature(object = "MEDMEset"): prints a summary of the object content  
**smoothed** signature(object = "MEDMEset"): extracts the Absolute Methylation Score from the AMS slot

## Author(s)

Mattia Pelizzola

## References

<http://genome.cshlp.org/cgi/content/abstract/gr.080721.108v1>

## See Also

[MEDME.readFiles](#), [MEDME.writeFiles](#)

## Examples

```
showClass("MEDMEset")
```

---

smooth

*Determining weighted MeDIP data*

---

## Description

MeDIP data from tiling arrays are smoothed by determining for each probe *i* the weighted average of the probes within a window of size *wsiz*e centered at *i*

## Usage

```
smooth(data, wsiz=1000, wFunction='linear')
```

## Arguments

<b>data</b>	An object of class MEDMEset
<b>wsiz</b>	number; the size of the smoothing window, in bp
<b>wFunction</b>	string; the type of weighting function, to choose among linear, exp, log or none

## Details

The un-smoothed data are read from the slot logR of the data MEDMEset and the resulting smoothed data are saved on the smoothed slot.

## Value

An object of class MEDMEset. In particular, the smoothed data are saved on the smoothed slot.

## Examples

```
data(testMEDMEset)
# just an example with the first 1000 probes
testMEDMEset = smooth(data = testMEDMEset[1:1000,])
```

---

testMEDMEset

*Dataset of class MEDMEset for testing MEDME*


---

## Description

This dataset contains a subset of the data reported in references. It contains normalized un-smoothed probe-level MeDIP enrichment for almost 50000 probes. This is a random subset of a custom Nimblegen chromosome X tiling array. It is a two channels array with an resolution of 100bp and oligos of 60nt. The M value is reported only. The fullyMet column of the logR slot contains data from a calibration experiments where MeDIP has been applied to a fully methylated sample. The last two columns NBME1 and YUSAC2 contain DNA-methylation experimental data for two cell strains: NBME1 are newborn normal melanocytes cells and YUSAC2 a melanoma strain. Data was processed with within and between array normalization. The full dataset contains almost 380K probes. See references for details. Chromosome, genomic position and logR of probes can be accessed with the methods chr, pos and logR respectively.

Please note that the original genomic coordinates were mapped to the hg17 human genome. These have been converted to hg18 using the LiftOver UCSC tool available online for batch conversion.

## Usage

```
data(testMEDMEset)
```

## Format

MEDMEset

## References

<http://genome.cshlp.org/cgi/content/abstract/gr.080721.108v1>

# Index

## \* classes

- MEDMEset-class, [6](#)
- [,MEDMEset-method (MEDMEset-class), [6](#)
- AMS (MEDMEset-class), [6](#)
- AMS, MEDMEset-method (MEDMEset-class), [6](#)
- CG (MEDMEset-class), [6](#)
- CG, MEDMEset-method (MEDMEset-class), [6](#)
- CGcount, [2](#), [3](#), [4](#)
- CGcounts (MEDMEset-class), [6](#)
- chr (MEDMEset-class), [6](#)
- chr, MEDMEset-method (MEDMEset-class), [6](#)
- initialize, MEDMEset-method  
(MEDMEset-class), [6](#)
- logR (MEDMEset-class), [6](#)
- logR, MEDMEset-method (MEDMEset-class), [6](#)
- MEDME, [3](#), [4](#)
- MEDME.predict, [4](#), [6](#)
- MEDME.readFiles, [5](#), [6](#), [7](#)
- MEDME.writeFiles, [5](#), [6](#), [7](#)
- MEDMEset-class, [6](#)
- org (MEDMEset-class), [6](#)
- org, MEDMEset-method (MEDMEset-class), [6](#)
- organism (MEDMEset-class), [6](#)
- pos (MEDMEset-class), [6](#)
- pos, MEDMEset-method (MEDMEset-class), [6](#)
- RMS (MEDMEset-class), [6](#)
- RMS, MEDMEset-method (MEDMEset-class), [6](#)
- show, MEDMEset-method (MEDMEset-class), [6](#)
- smooth, [2-4](#), [7](#)
- smoothed (MEDMEset-class), [6](#)
- smoothed, MEDMEset-method  
(MEDMEset-class), [6](#)
- testMEDMEset, [8](#)